

# Generative Adversarial Network

בס"ד

Presenting:

**Ben Remez**

**Barack Samuni**

**Omri Sason**

**Barak Yaakov**

# Generative Adversarial Network

**1** Network Architecture

**2** Project Goal

**3** Data Description

**4** Data Preprocessing

**5** Model Progression

**6** The Best Model

**7** The Experiments

**8** Results

**9** Conclusions

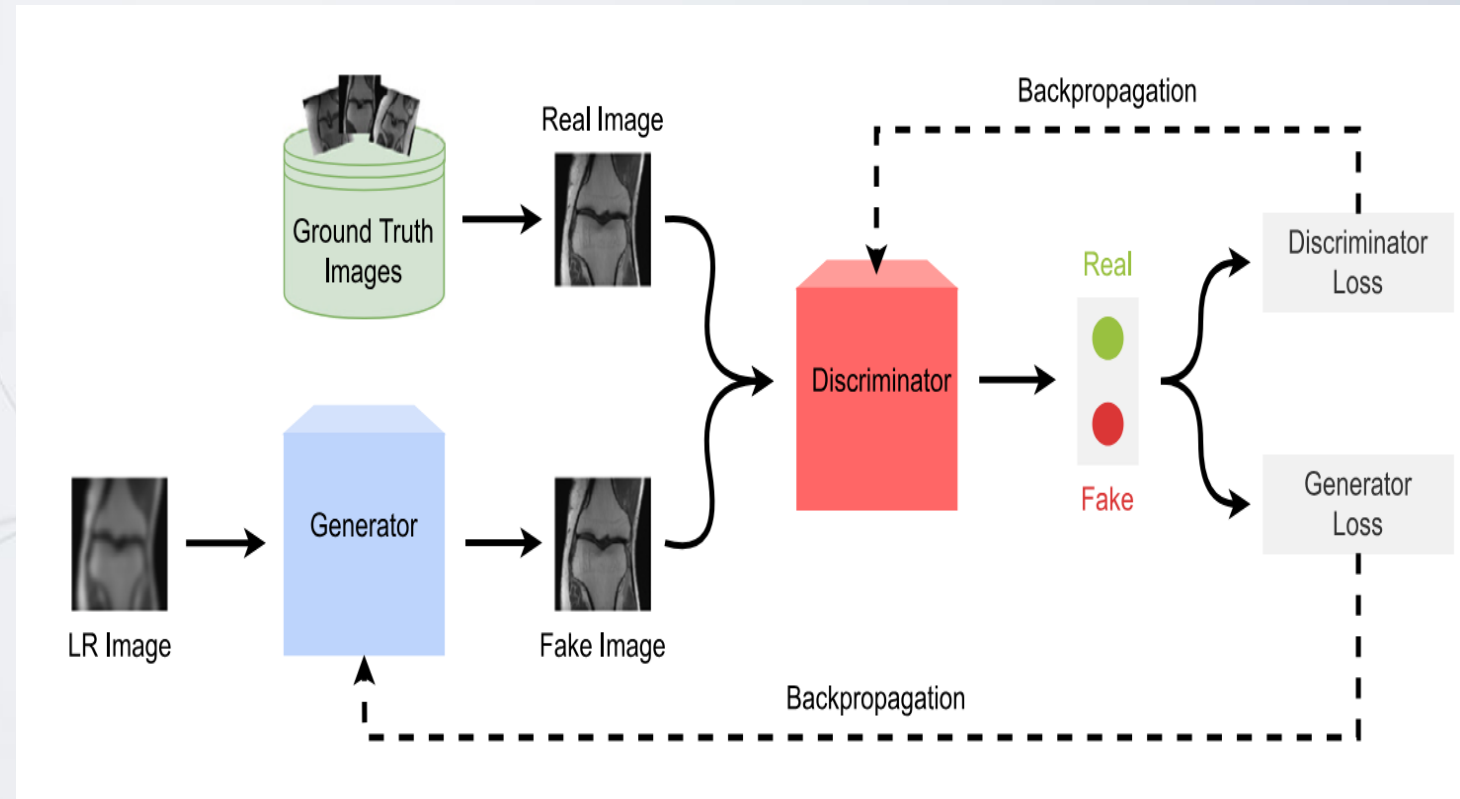
**10** Discussions and Future Work

**11** Final Summary

# Generative Adversarial Network (GAN)

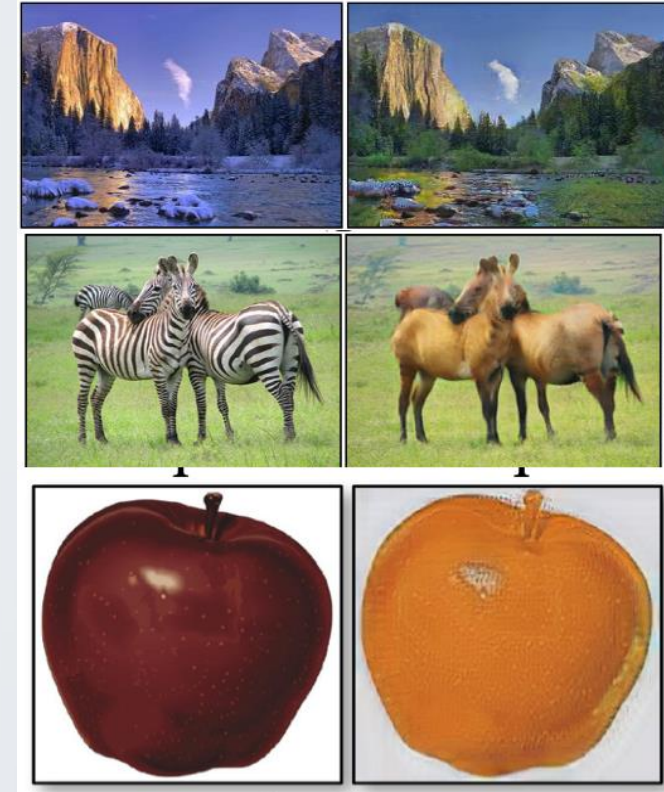
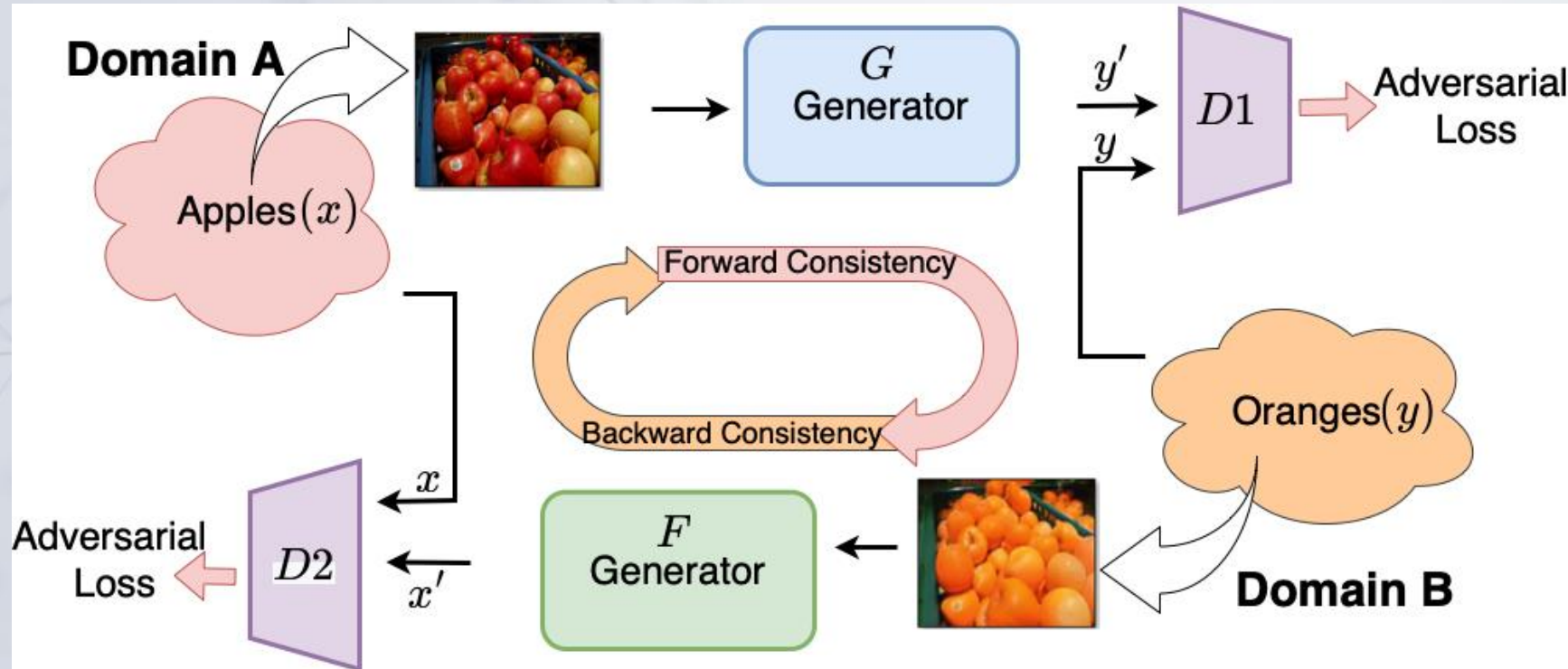
## How do we **TRAIN** a Generative Adversarial Network?

- Discriminator- The GAN also consists of a discriminator, that aims to distinguish between fake generated images and real ones.
- Generator- the generator trains to “fool” the Discriminator into thinking that his fake generated images are real ones.
- Meanwhile, the discriminator trains to be better in identifying the fake images.



# Cycle GAN

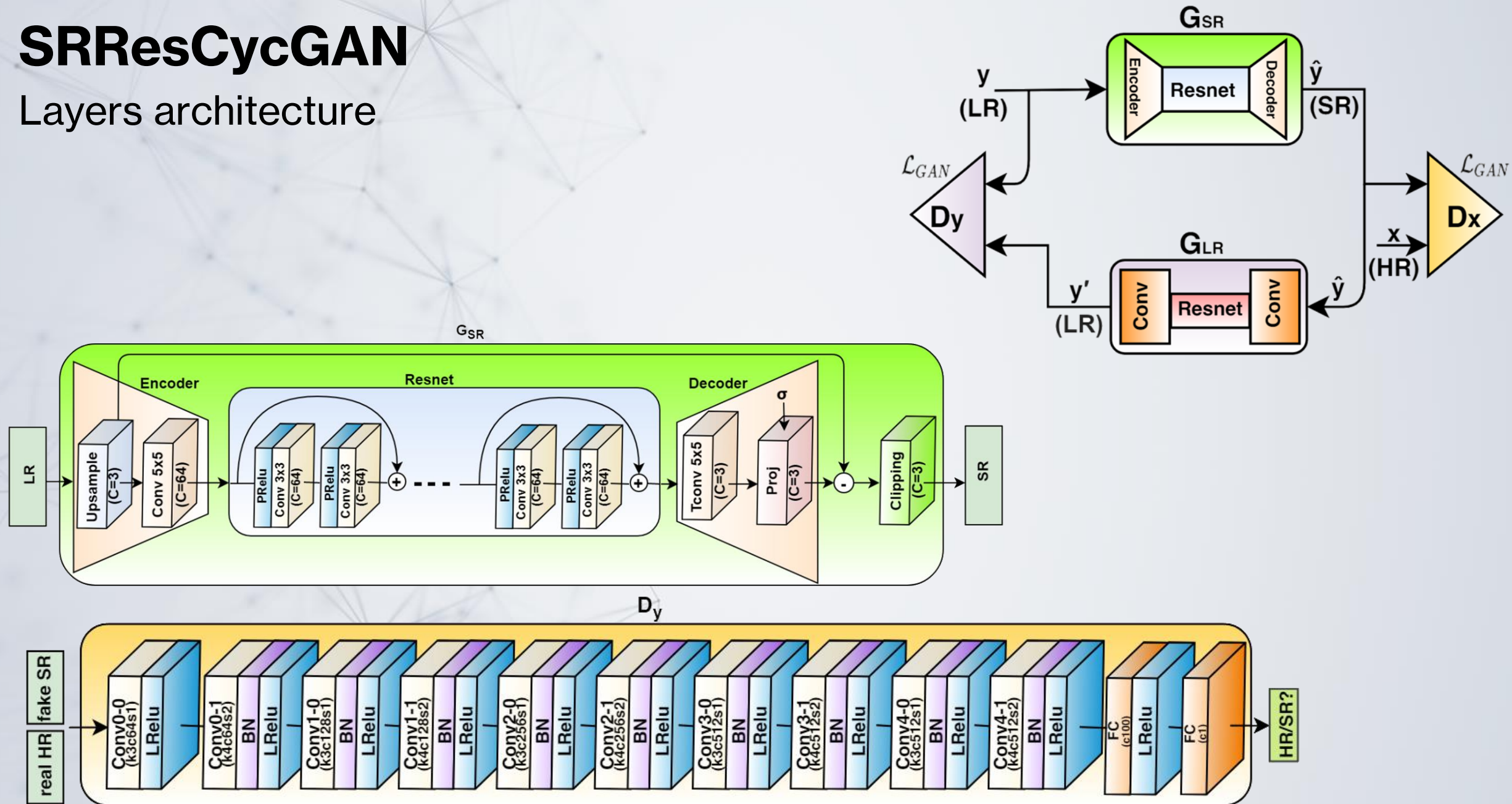
## Network architecture





# SRResCycGAN

## Layers architecture



# The Problem



- Image enhancement to low resolution images.
- Add distortion to images and construct high resolution images by mitigating the distortion.

# Project Goals



- Develop deep learning model to perform super-resolution tasks
- Improving the metrics and reducing losses by enhancing the performance of our model
- Introduce artificial distortion \ noise to images and then apply denoising techniques to reconstruct clean, high-quality images.

# Data Description



## DIV2K Dataset Overview

- **Purpose:** Image super-resolution tasks
- **Content:** 1,000 high-resolution images
- **Splits:**
  - 800 training images
  - 100 validation images
  - 100 test images
- **Features:** Diverse scenes and objects
- **Resolutions:** Multiple resolutions per image



# Data Description

## US License Plates Dataset Overview

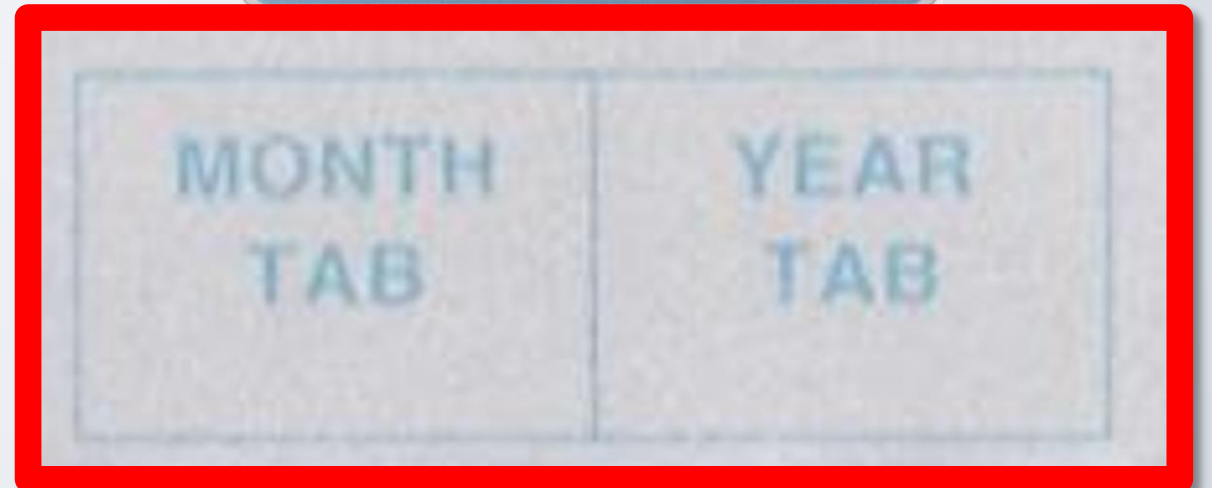
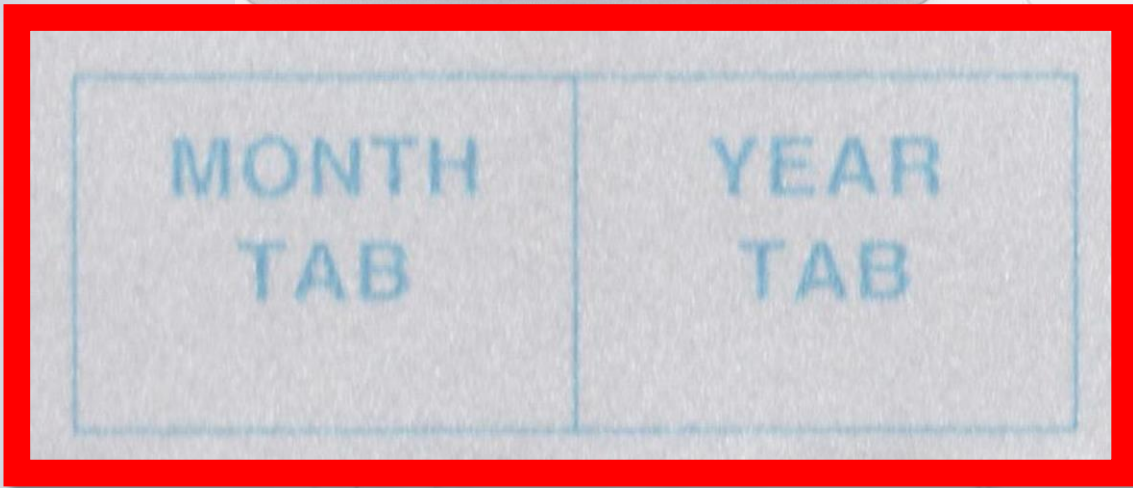
- **Purpose:** Classification tasks
- **Content:** 4,462 images
- **Splits after cleaning:**
  - 750 training images
  - 108 test images
- **Features:** Diverse US License Plates



# Data Preprocessing

## Downsampling

The image is resized by the specified scale factor (4 in our case) using the BICUBIC DISTORTION algorithm, which preserves more details in the downscaled image.



# Data Preprocessing

## Dimensional Distortion

- Computes a **rotation matrix calculation** to simulate a spatial rotation around the Y-axis.
- For each image, generating several spatial rotated versions - random angles 30-70 [Deg]





# Training The Model- Hyper-Parameters

## **Learning Rate**

Controls step size in training. Lower LR helps in fine-tuning for stability.

## **Batch Size**

Affects training stability and memory usage. Larger sizes give stable gradients.

## **Optimizer**

Influences convergence speed and efficiency. Adam is generally a safe choice

## **Number of Epochs**

Determines how long the model trains. Balance to avoid underfitting or overfitting.

# Our Paper

## Deep Generative Adversarial Residual Convolutional Networks for Real-World Super-Resolution

Rao Muhammad Umer    Gian Luca Foresti    Christian Micheloni

University of Udine, Italy.

engr.raoumer943@gmail.com, {gianluca.foresti, christian.micheloni}@uniud.it

### Abstract

*Most current deep learning based single image super-resolution (SISR) methods focus on designing deeper / wider models to learn the non-linear mapping between low-resolution (LR) inputs and the high-resolution (HR) outputs from a large number of paired (LR/HR) training data. They usually take as assumption that the LR image is a bicubic down-sampled version of the HR image. However, such degradation process is not available in real-world settings i.e. inherent sensor noise, stochastic noise, compression artifacts, possible mismatch between image degradation process and camera device. It reduces significantly the performance of current SISR methods due to real-world image corruptions. To address these problems, we propose a deep Super-Resolution Residual Convolutional Generative Adversarial Network (SRResCGAN<sup>1</sup>) to follow the real-world degradation settings by adversarial training the model with*



GT

ESRGAN

ESRGAN-FS

SRResCGAN

SRResCGAN+



# The Experiment From the Article

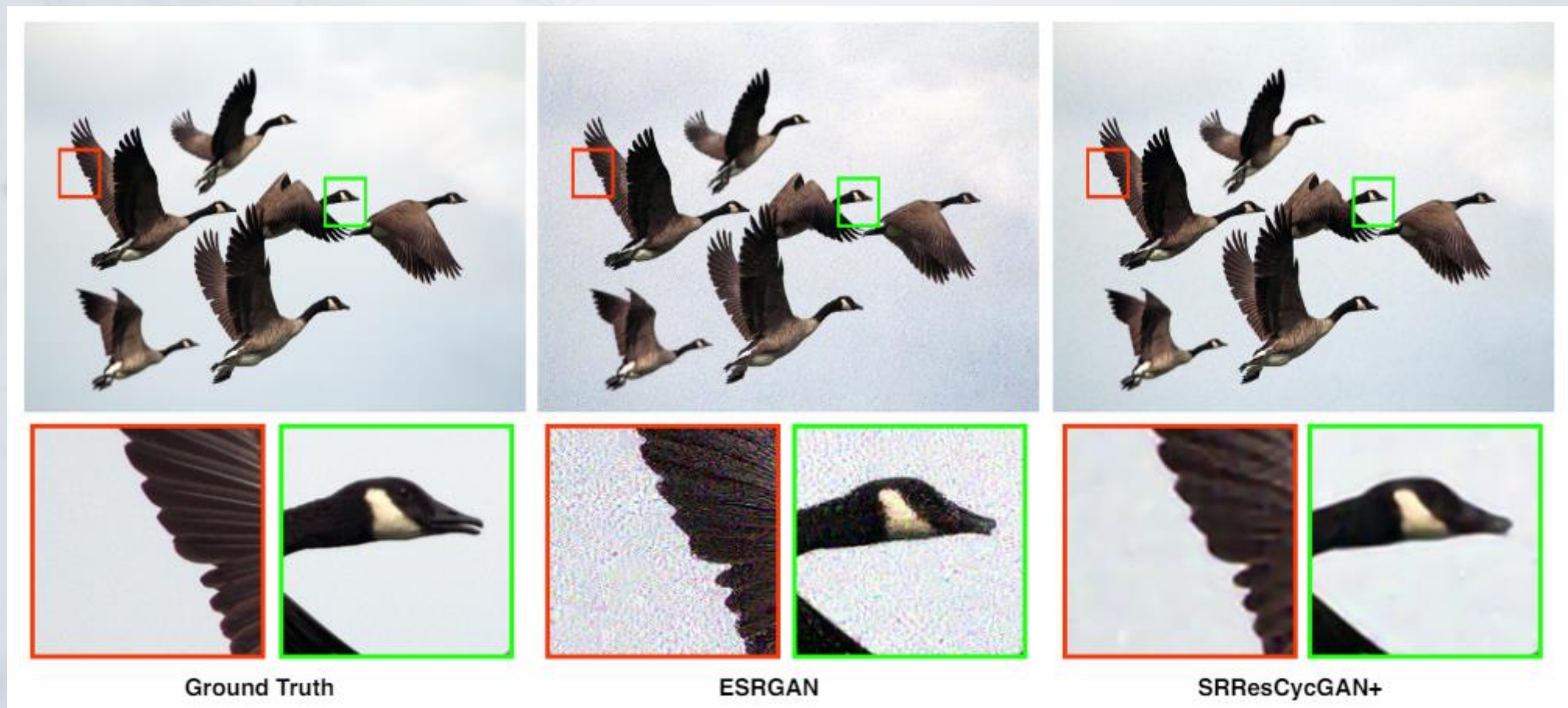
## Objective:

- Develop a super-resolution method that works well on real-world images, addressing the domain gap between low-resolution and high-resolution images.

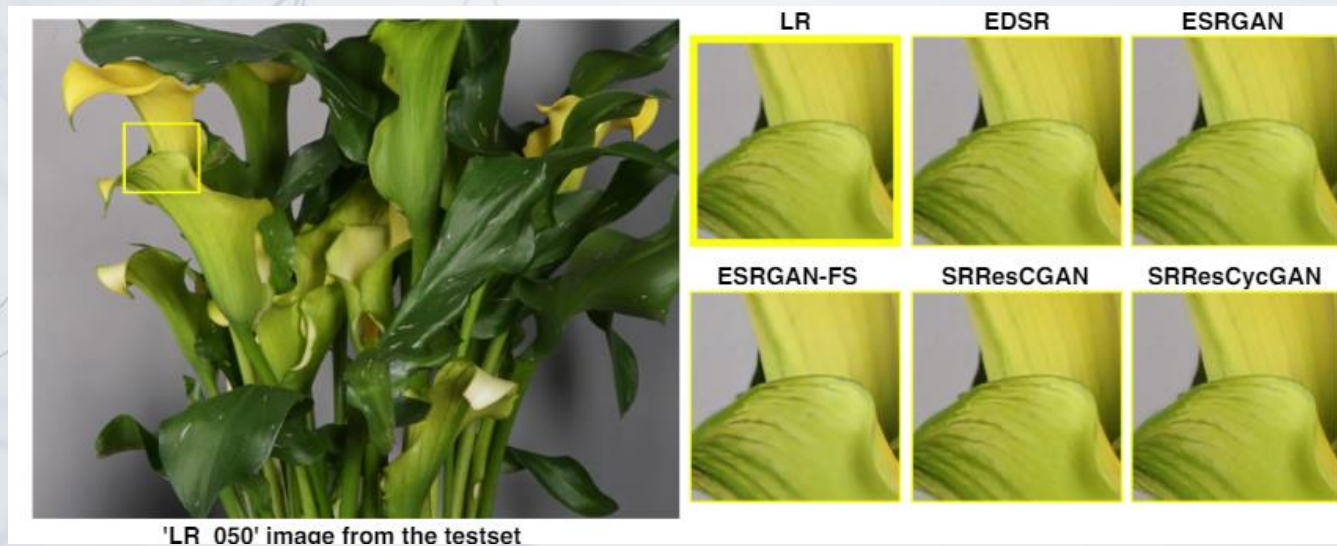
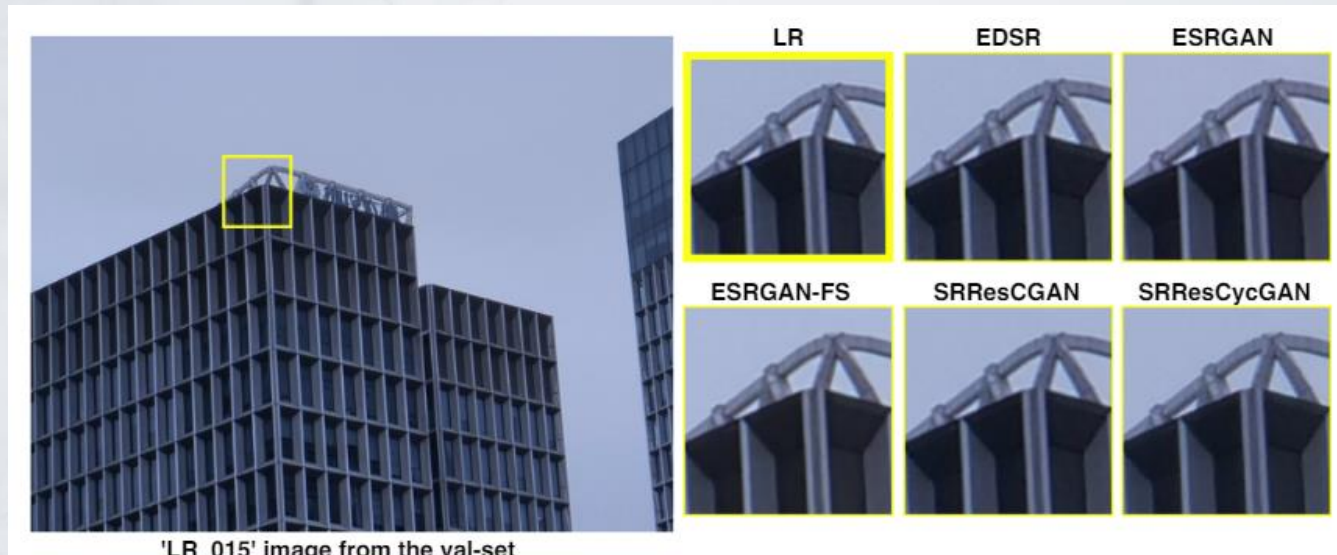
## Training:

- **Data:**
  - 2650 high-resolution images with synthetic degradations (noise, compression artifacts).
  - 800 clean high-resolution images from DIV2K dataset.
  - Additional 19000 low-resolution and high-resolution images from AIM2020 Real Image SR Challenge.
- **Loss Functions:**
  - Perceptual Loss: Ensures perceptual quality.
  - GAN Loss: Focuses on high-frequency details.
  - Content Loss (L1): Maintains content fidelity.
  - Total-Variation Loss: Enhances image sharpness.
  - Cyclic Loss: Ensures consistency between low-resolution and high-resolution images.
- **Optimizers:**
  - Adam optimizer with specific parameters.
  - Learning rate adjustments after specific iterations.

# Model results in article



# Model results in article





# The Experiments

## Experiment 1

Training on 750 images of license plates without any distortion and without using a pre-trained dataset.



## Experiment 2

Training on 800 images from the DIV2K dataset without any distortion using a pre-trained model based on license plates.



+



## Experiment 3

Training on 750 distorted images of license plates) dimensional distortion (using a pre-trained model based on license plates.



+



## Experiment 4

Trained on 750 distorted images of license plates (dimensional distortion) using a pre-trained model based on both license plates and the DIV2K dataset.



+



+











## Experiment 5

Training the model from the Article on 500 images of license plates without any distortion ,according to the setting described in the article



# Training The Model

<u>Train</u>	<u>Dataset</u>	<u>Number of images</u>	<u>Epochs</u>	<u>Generator Learning Rate</u>	<u>Discriminator Learning Rate</u>	<u>SR Output</u>
1	Car plates	88	10	$10^{-4}$	$10^{-4}$	
2	Car plates	150	10	$10^{-4}$	$10^{-4}$	
3	Car plates	150	10	$1 \times 10^{-3}$	$10^{-4}$	
4	Car plates	150	10	$2 \times 10^{-4}$	$5 \times 10^{-5}$	
5	Car plates	150	30	$2 \times 10^{-4}$	$5 \times 10^{-5}$	
6	Car plates	1000	5	$2 \times 10^{-4}$	$5 \times 10^{-5}$	
7	Car plates	3600	30	$2 \times 10^{-4}$	$5 \times 10^{-5}$	
8	Car plates	3600	30	Starting with $1 \times 10^{-5}$ Going down by factor of 10 every 10 epochs	Starting with $1 \times 10^{-5}$ Going down by factor of 10 every 10 epochs	



# Training The Model 3 Final

<u>Train</u>	<u>Dataset</u>	<u>Number of images</u>	<u>Epochs</u>	<u>Generator Learning Rate</u>	<u>Discriminator Learning Rate</u>	<u>Loaded Train Model No. Weights</u>	<u>Fine Tuning</u>
<b>9</b>	Car Plates	750	30	$2 \times 10^{-4}$	$1 \times 10^{-5}$	X	X
<b>10</b>	DIV2K	800	30	$2 \times 10^{-4}$	$1 \times 10^{-5}$	9	X
<b>11</b>	Car Plates	750	30	Changing learning rate starting with $1 \times 10^{-5}$	X	10	V

# Training The Model (Distortion)

<u>Train</u>	<u>Dataset</u>	<u>Number of images</u>	<u>Epochs</u>	<u>Generator Learning Rate</u>	<u>Discriminator Learning Rate</u>	<u>Loaded Train Model No. Weights</u>	<u>Fine Tuning</u>
<b>12</b>	Car Plates Distorted	750	30	Changing learning rate starting with $1 \times 10^{-5}$	X	9	V
<b>13</b>	Car Plates Distorted	750	30		X	10	V
<b>14</b>	Car Plates Distorted (HR)	750	30	$2 \times 10^{-4}$	$1 \times 10^{-5}$	X	X

# Network Loss

$$\mathcal{L}_{G_{SR}} = \mathcal{L}_{\text{per}} + \mathcal{L}_{\text{GAN}} + \mathcal{L}_{tv} + 10 \cdot \mathcal{L}_1 + 10 \cdot \mathcal{L}_{cyc}$$

## Perceptual Loss

Sharpness, color, contrast, distortion, texture, resolution, perspective, patterns

$$\mathcal{L}_{\text{per}} = \frac{1}{N} \sum_i \mathcal{L}_{\text{VGG}} = \frac{1}{N} \sum_i \|\phi(\mathbf{G}_{SR}(\mathbf{y}_i)) - \phi(\mathbf{x}_i)\|_1$$

## Texture Loss

Quality of generator and discriminator

$$\mathcal{L}_{\text{GAN}} = \mathcal{L}_{\text{RaGAN}} = -\mathbb{E}_{\mathbf{x}} [\log (1 - \mathbf{D}_{\mathbf{x}}(\mathbf{x}, \mathbf{G}_{SR}(\mathbf{y})))] - \mathbb{E}_{\hat{\mathbf{y}}} [\log (\mathbf{D}_{\mathbf{x}}(\mathbf{G}_{SR}(\mathbf{y}), \mathbf{x}))]$$

## Content Loss

High level features difference between generator and GT

$$\mathcal{L}_1 = \frac{1}{N} \sum_i \|\mathbf{G}_{SR}(\mathbf{y}_i) - \mathbf{x}_i\|_1$$

## Total Variation Loss

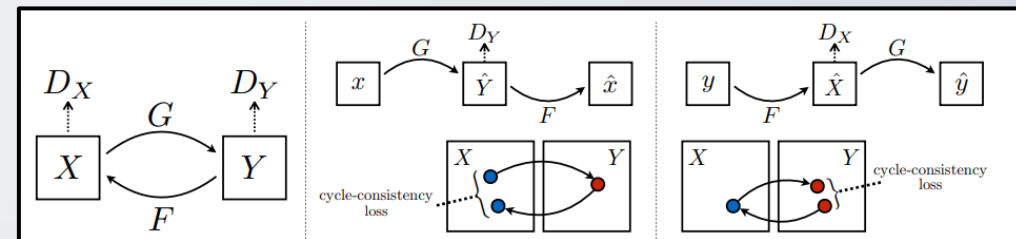
Resolution difference between generator and GT

$$\mathcal{L}_{tv} = \frac{1}{N} \sum_i (\|\nabla_h \mathbf{G}_{SR}(\mathbf{y}_i) - \nabla_h(\mathbf{x}_i)\|_1 + \|\nabla_v \mathbf{G}_{SR}(\mathbf{y}_i) - \nabla_v(\mathbf{x}_i)\|_1)$$

## Cyclic Loss

CycleGan consistency

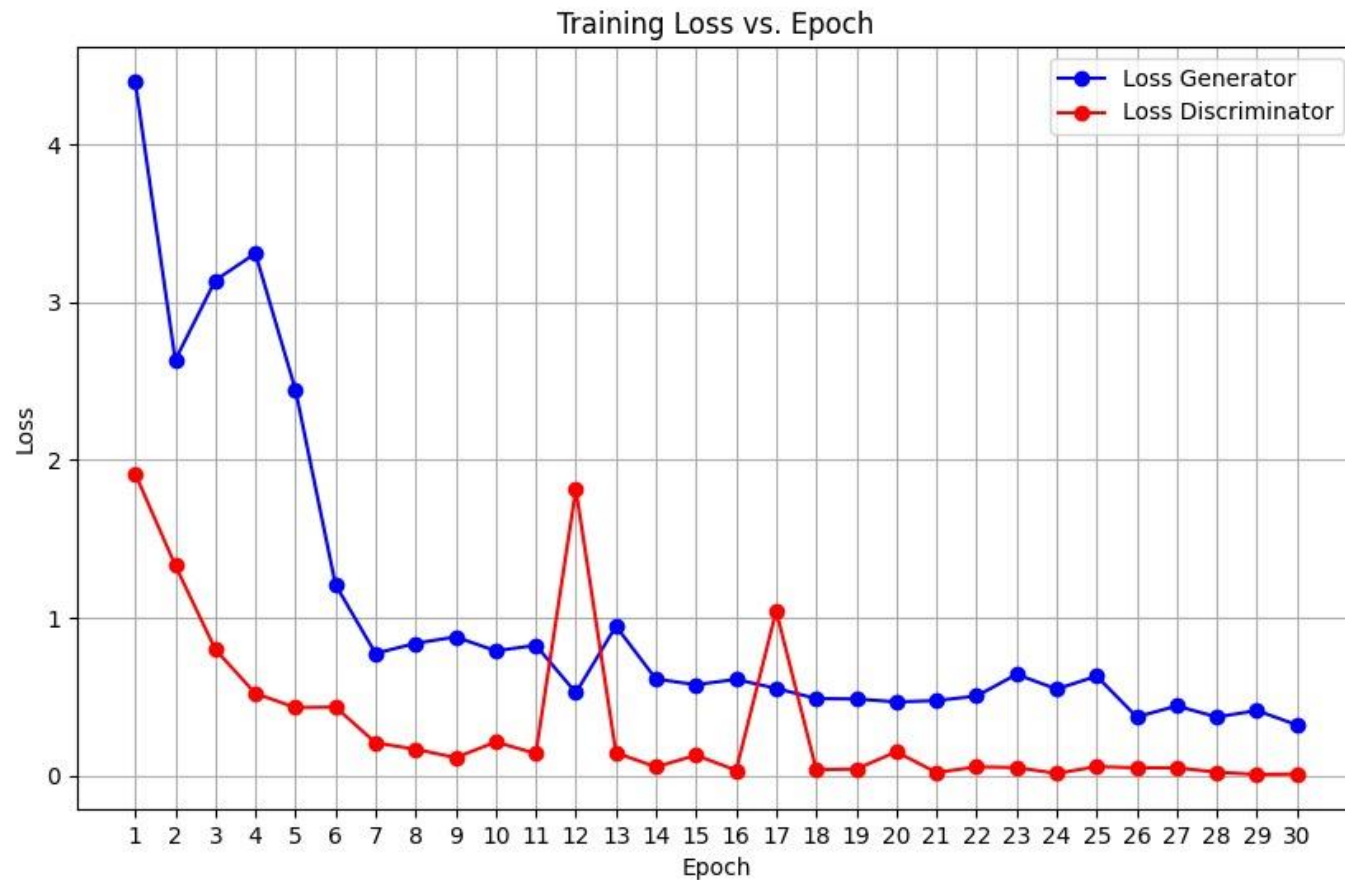
$$\mathcal{L}_{cyc} = \frac{1}{N} \sum_i \|\mathbf{G}_{LR}(\mathbf{G}_{SR}(\mathbf{y}_i)) - \mathbf{y}_i\|_1$$



# Pretrain : No Pretrain

## Train : License Plates Dataset

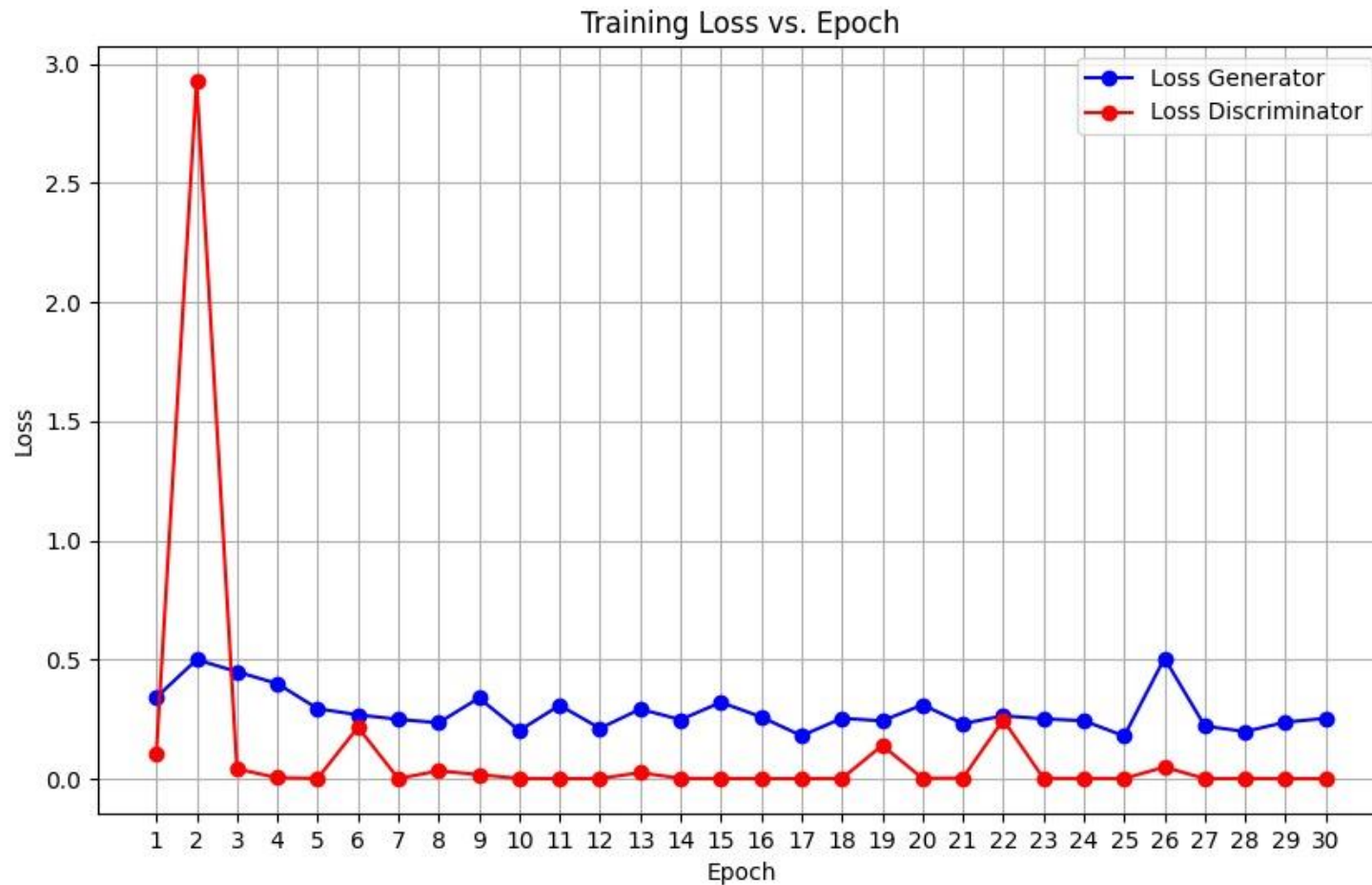
### Without Distortion



# Pretrain : License Plates Dataset

## Train : DIV2K Dataset

### Without Distortion



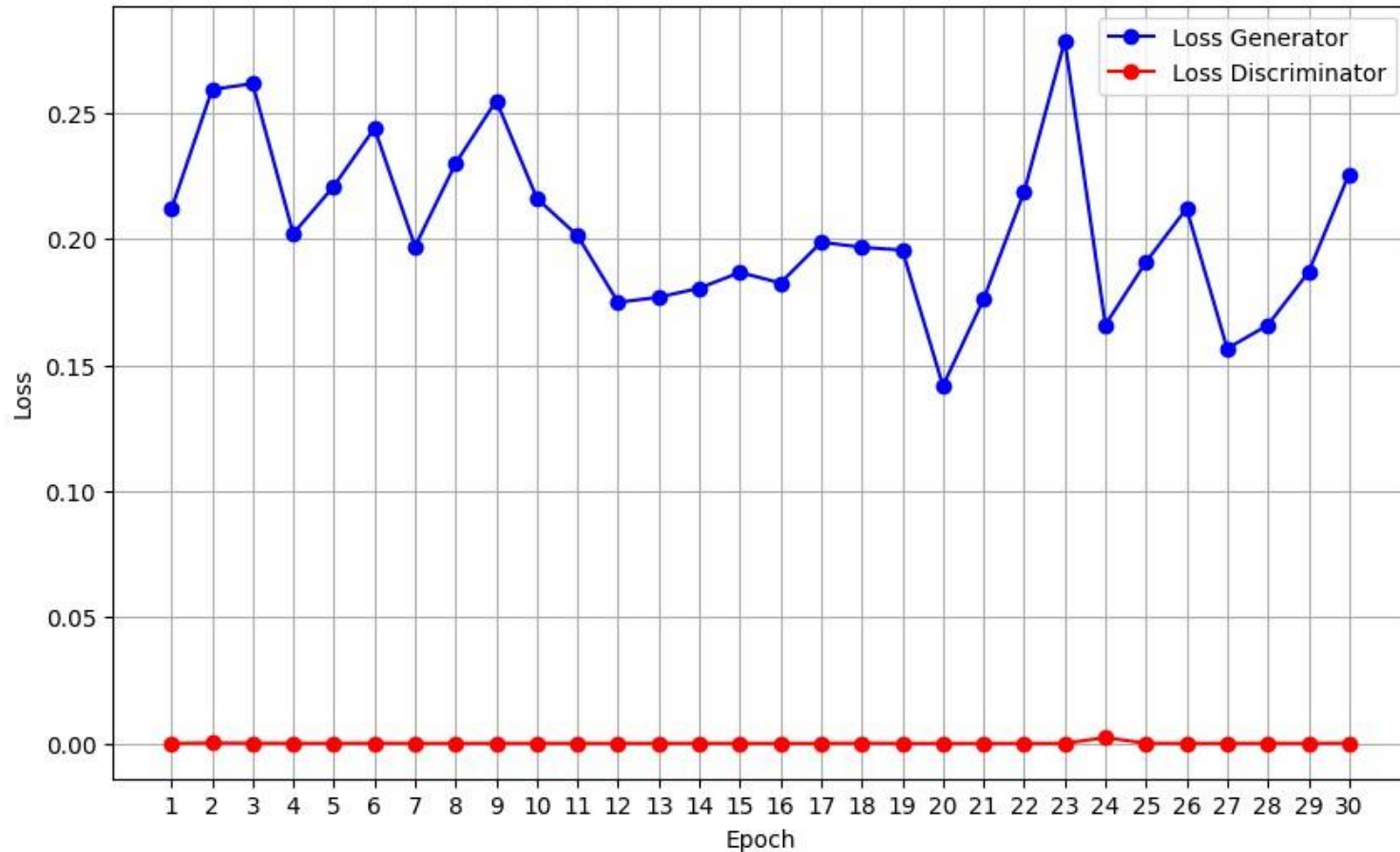


# Pretrain : DIV2K

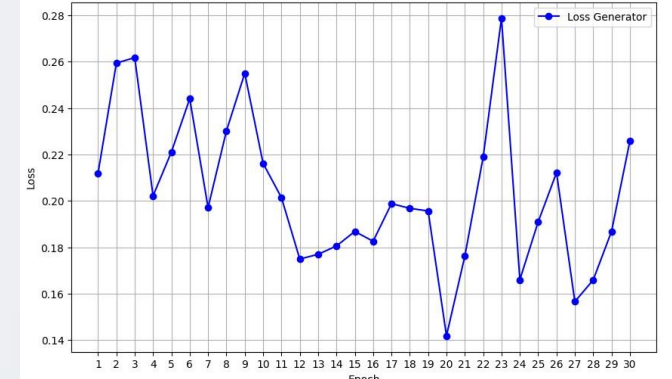
## Train : License Plates Dataset

### Distortion: Dimensional

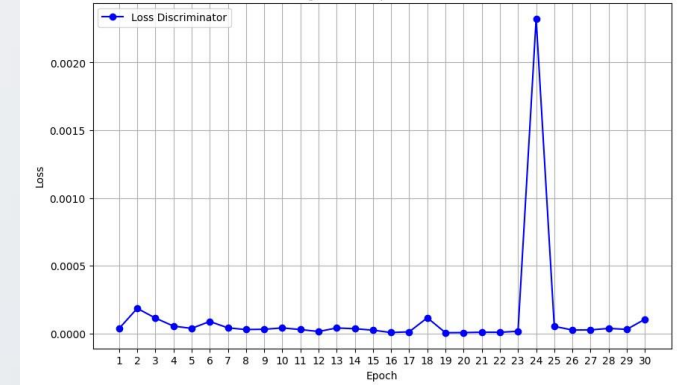
Training Loss vs. Epoch



Training Loss vs. Epoch Generator

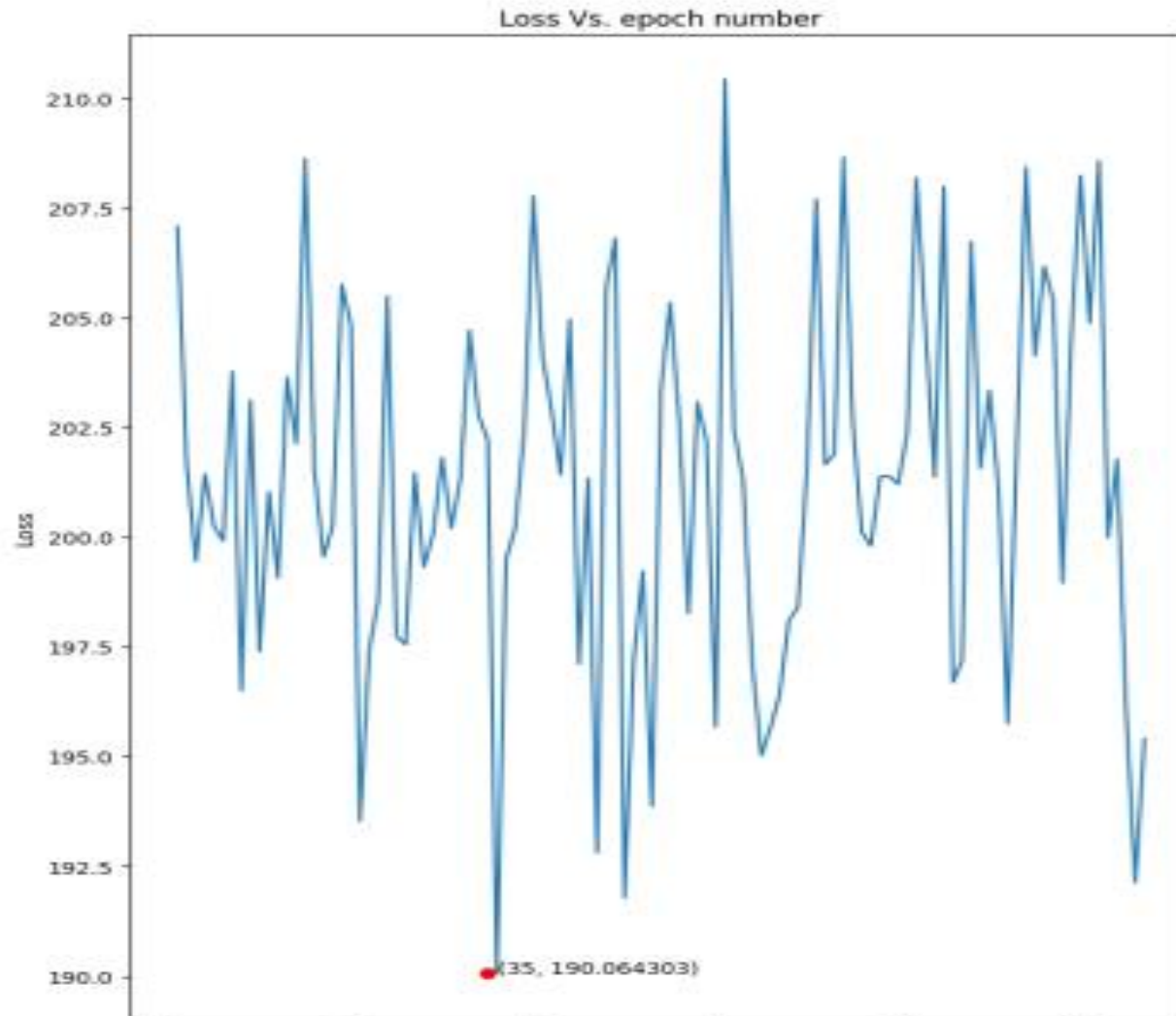


Training Loss vs. Epoch Discriminator



# Article's Model

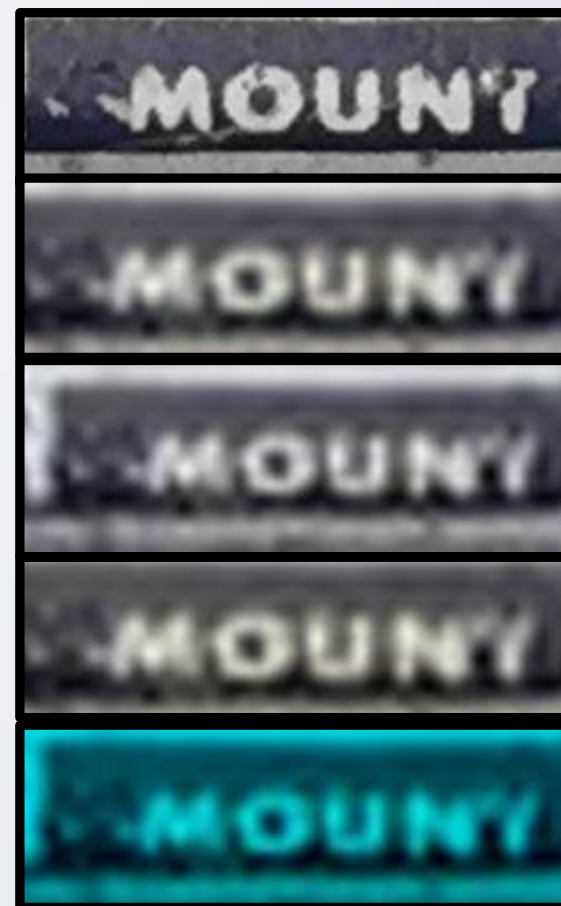
**Pretrain : G-perceptual model weight**  
**Train : License Plates Dataset**  
**Without Distortion**



# Model Progression

All Results are at x180 zoom

GT



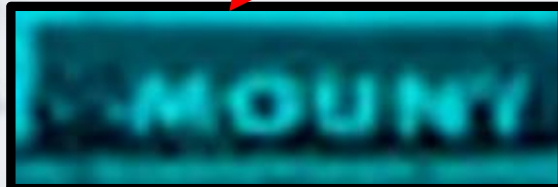
GT

Epoch 30

Epoch 20

Epoch 10

Epoch 5



**Epoch 5**



**Epoch 10**



**Epoch 20**



**Epoch 30**



**GT**



**LR**



**X800**



**SR**



**X160**



**GT**



**X160**



**Train : License Plates Dataset**





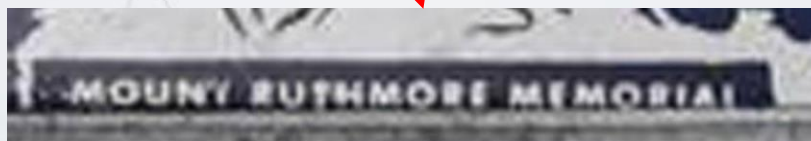
**LR**

**X800**



**SR**

**X260**



**GT**

**x260**



**Pretrain : License Plates Dataset**  
**Train : DIV2K Dataset**

**LR**



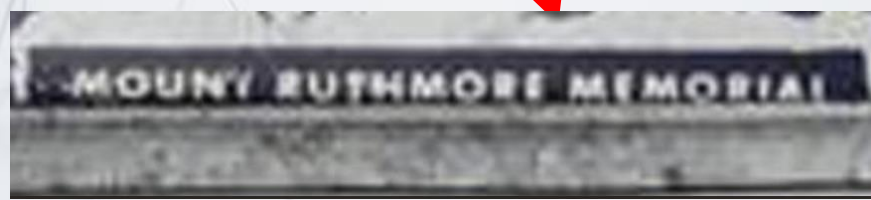
**X800**



**SR**



**X160**



**GT**



**X160**



**Pretrain: License Plates Dataset + DIV2K Dataset**  
**Fine Tune: License Plates Dataset**



# Distortion Models Outputs

**Pretrain:** License Plates + DIV2K Datasets

**Fine Tune:** License Plates Distorted

**Pretrain:** License Plates Dataset

**Fine Tune:** License Plates Distorted



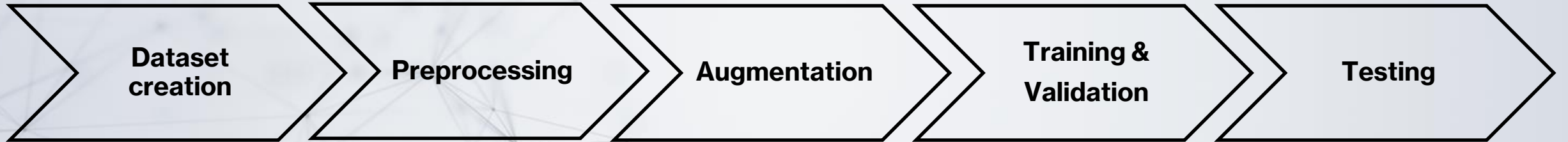
GT



LR



# Article Model



# Dataset creation - DSGAN

<u>The Problem</u>	<u>The Solution - DSGAN</u>	<u>Application in Our Model</u>
Super-resolution tasks usually assume the LR image is simply a downsampled version of the HR image.	DSGAN is a specialized network designed to address such deformations.	Our model integrates DSGAN to generate pairs of HR images and deformed LR images from the original input pairs.
In reality, LR images may suffer from deformations like geometric transformations and misalignments, affecting the quality of super-resolution.	It estimates and corrects deformations before applying super-resolution, enhancing the model's realism and robustness.	This training method enables the model to handle deformations effectively and perform denoising, making it more adaptable to real-world super-resolution challenges.



# Dataset creation - DSGAN



# Preprocessing-Data repetition

## Dataset Repeated x3: Each Image Trained 3 Times

### Advantages

- **Improved stability in training**

Repeated data provides smoother gradients, aiding convergence in smaller datasets.

- **Increased dataset size**

Increases dataset size, helping the model learn more robust features, especially with small datasets.

### Disadvantages

- **Overfitting Risk:**

Repeated data can cause the model to memorize rather than generalize, leading to overfitting

- **Increased Runtime**

More data to process results in longer training times, but not necessarily better learning.

- **Higher Memory Consumption-**

Data repetition can increase memory consumption, straining RAM or GPU resources..

- **Learning efficiency-**

Frequent repetition of examples may slow down the model's learning progress

# Data augmentation-Data mixup

## Linear Mix: Each Image Combined with a Random Permutation

### Advantages of Mixup

#### Improved Generalization

Creates smoother decision boundaries, reducing overfitting, especially with limited data.

#### Enhanced Detail Reconstruction

Improves recovery of fine textures and gradients, leading to higher-quality images.

#### Robustness to Noise

Increases the model's resilience to noise and content variations, enhancing performance across datasets.

#### Balanced Resolution Learning

Aids the model in adapting to and learning from different resolutions.

### Disadvantages of Mixup

#### Increased Computational Overhead

Raises training time and complexity.

#### Potential Underfitting

May blur distinctions between images

#### Blurring of Fine Details

Can smooth out crucial details, affecting super-resolution quality.

#### Complex Hyperparameter Tuning

Finding optimal beta distribution parameters can be challenging and affect benefits.



# Optional Chopping the Image into patches

- The dataset contains high resolution images that sometimes are too hard for the model / GPU to process.
- In order to prevent CUDA from crashing, and handle memory consumption, the Image is recursively chopped into overlapping patches until it reaches a pre-defined minimal size and only then it passes through the network.
- Outputs are generated and the final output is being reassembled from all the outputs.
- In this way the GPU is not running out of memory when processing the Image.



# Training process

- The model is using a pre-trained network based on SR-RESGAN.
- The training is done on batches of 16 images.'
- 'the model uses AMSGrad optimizer for both Generator and Discriminator (which is an improvement for ADAM)
- The model trains for 10000 iterations, so epochs number is defined this way:
  - + The dataset we chose contains 500 images for training, but it is being repeated 3 times, so 1500 images is being trained with repetition.
  - + Each batch contains 16 mages
  - + It means that there are  $1500 / 16 = 93.75$  which is floored to 93 batches
  - + So the number of epochs should be:  $10000 / 93 = 107.52$  which is floored to 107.
- Learning rates are halved in iterations [5000, 10000, 20000, 30000]



**GROUND TRUTH , x68**

**SUPER RESOLUTION, x68**

**LOW RESOLUTION , x313**

**Article's Model**

**Pretrain : G-perceptual model weight**

**Train : DIV2K Dataset**

# PSNR

## Peak Signal-to-Noise Ratio (PSNR)

**Purpose:** Measures quality of reconstructed or compressed images/videos vs. original.

### Calculation:

- **MAX:** Maximum pixel value (255 for 8-bit images).
- **MSE:** Mean Squared Error between original and reconstructed images.

### Interpreting PSNR Values:

- **30 dB and above:** Good quality, minor visible differences.
- **20 - 30 dB:** Acceptable quality, noticeable distortions.
- **Below 20 dB:** Poor quality, significant visible distortions.

$$\text{MSE} = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2$$

$$\text{PSNR} = 10 \times \log_{10} \left( \frac{\text{MAX}^2}{\text{MSE}} \right)$$

# LPIPS

## Learned Perceptual Image Patch Similarity (LPIPS)

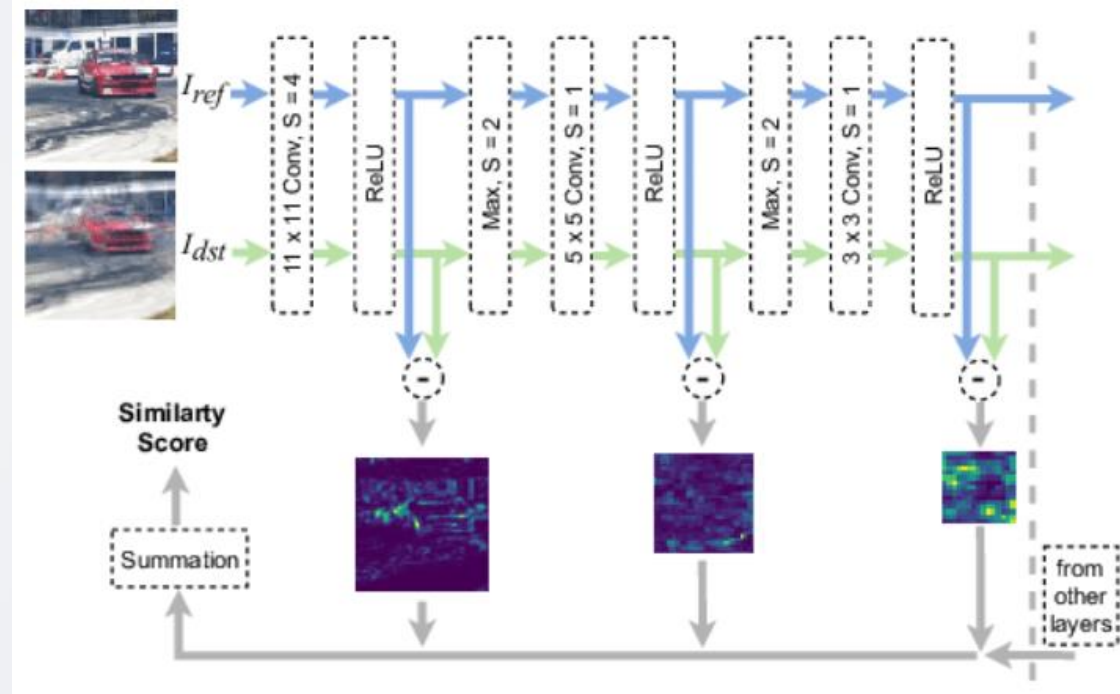
**Purpose:** Evaluates perceptual similarity between images using deep neural networks.

### Compared to Traditional Metrics:

- **PSNR & SSIM:** Focus on pixel-wise differences.
- **LPIPS:** Captures perceptual differences closer to human vision.

### Interpreting LPIPS Scores:

- **Lower Score:** Higher perceptual similarity (better quality).
- **Higher Score:** Greater perceptual difference (lower quality).





# SSIM

## Structural Similarity Index (SSIM)

**Purpose:** Measures image similarity by modeling human visual perception.

### Compared to Traditional Metrics:

- **MSE & PSNR:** Focus on pixel-wise differences.
- **SSIM:** Considers luminance, contrast, and structure.

### Interpreting SSIM Values:

- **SSIM = 1:** Images are identical.
- **SSIM > 0.9:** Images are highly similar.
- **SSIM 0.7 - 0.9:** Images are somewhat similar with visible differences.
- **SSIM < 0.7:** Images have noticeable differences.
- **SSIM ≤ 0:** Images are very different.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

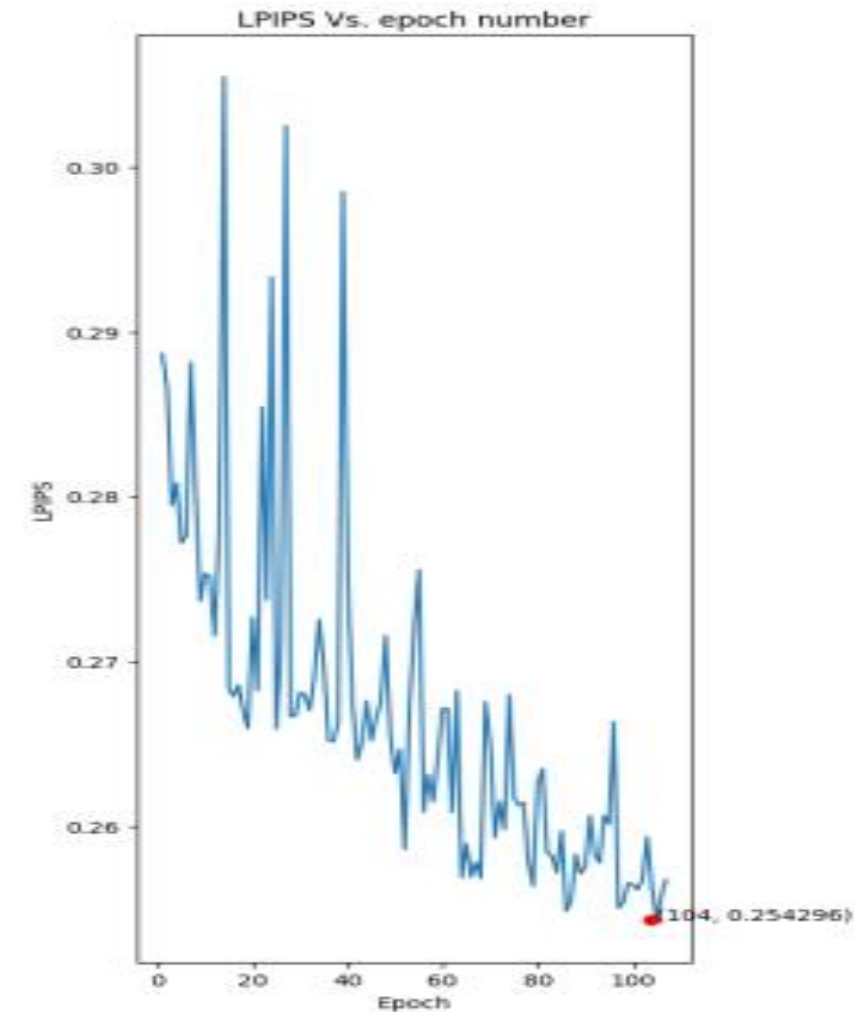
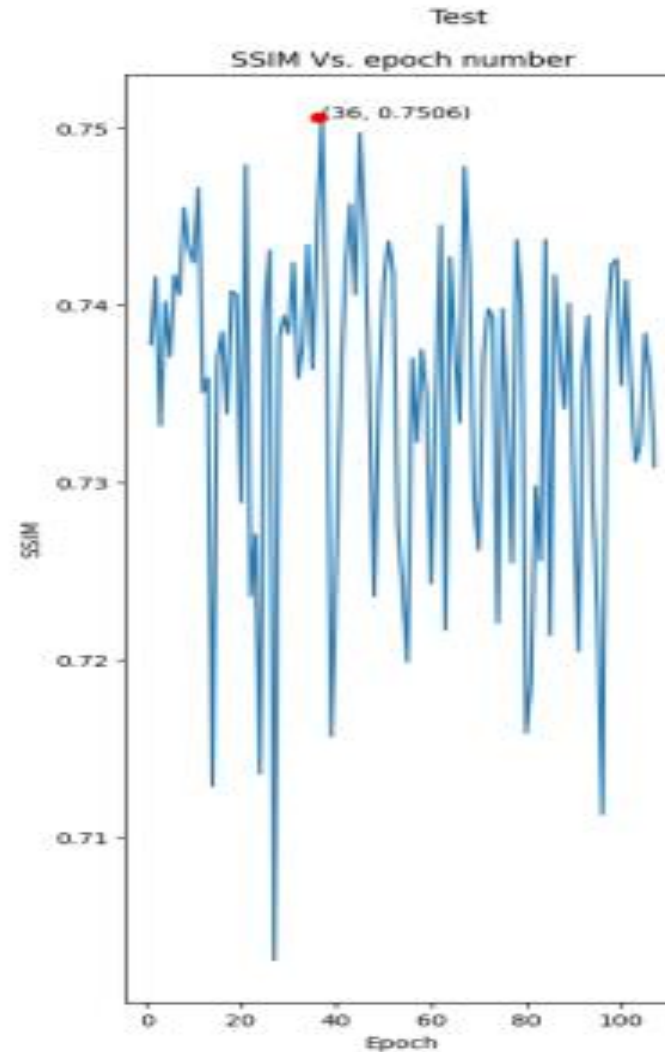
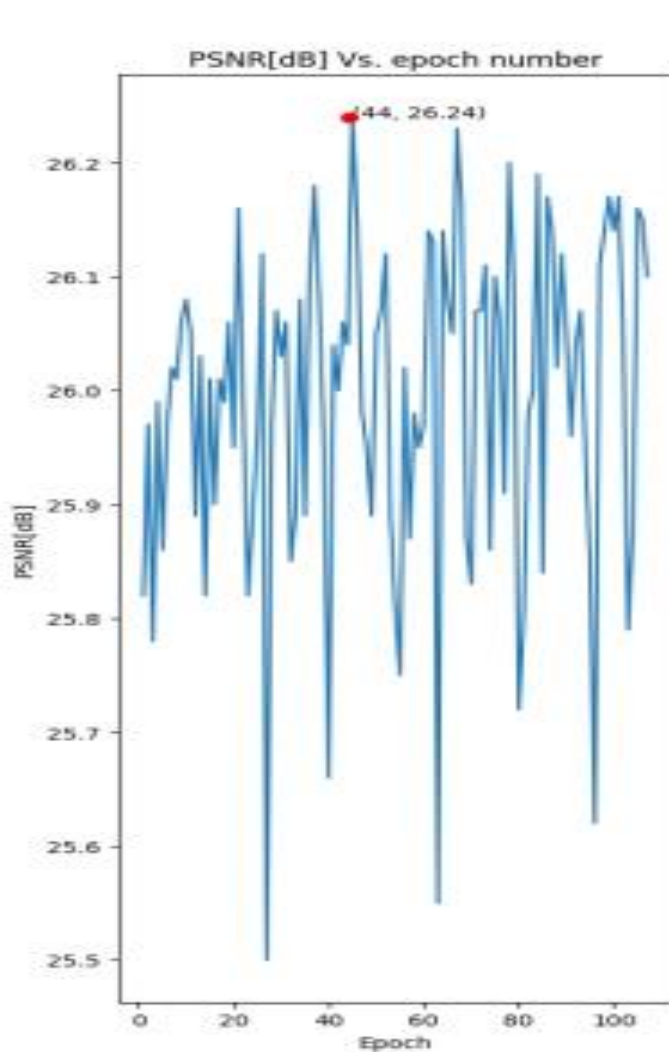
where:

- $\mu_x$  and  $\mu_y$  are the mean intensities of images  $x$  and  $y$ .
- $\sigma_x^2$  and  $\sigma_y^2$  are the variances of  $x$  and  $y$ .
- $\sigma_{xy}$  is the covariance between  $x$  and  $y$ .
- $C_1$  and  $C_2$  are constants to stabilize the division.

# Article's Model

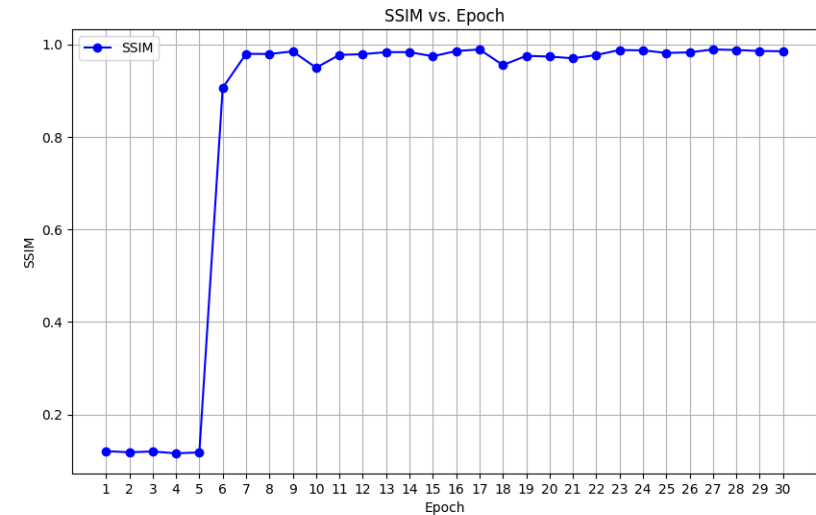
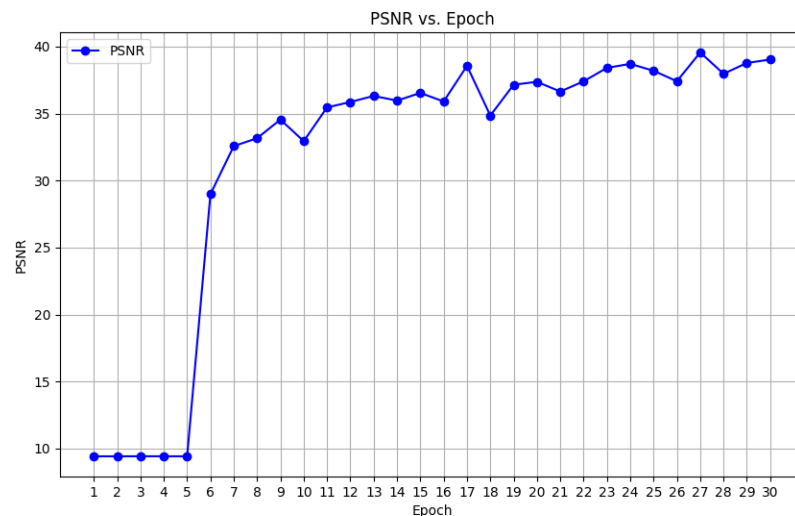
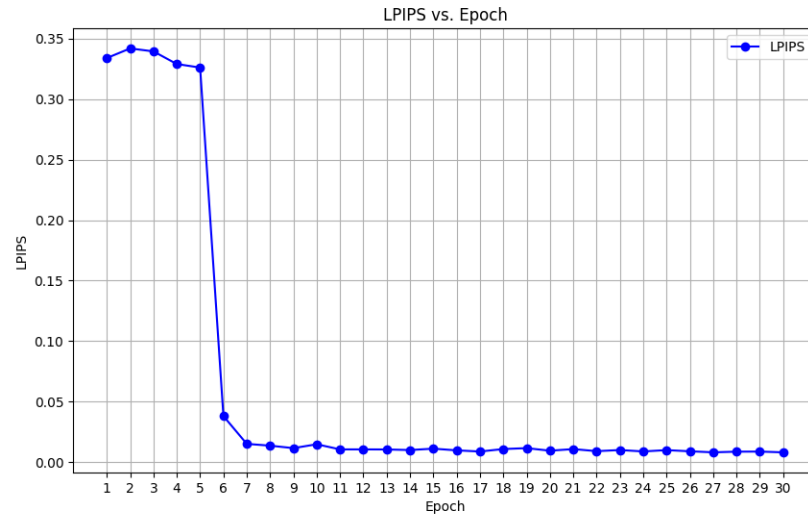
**Pretrain : G-perceptual model weights (datasets from article)**

**Train : License Plates Dataset**



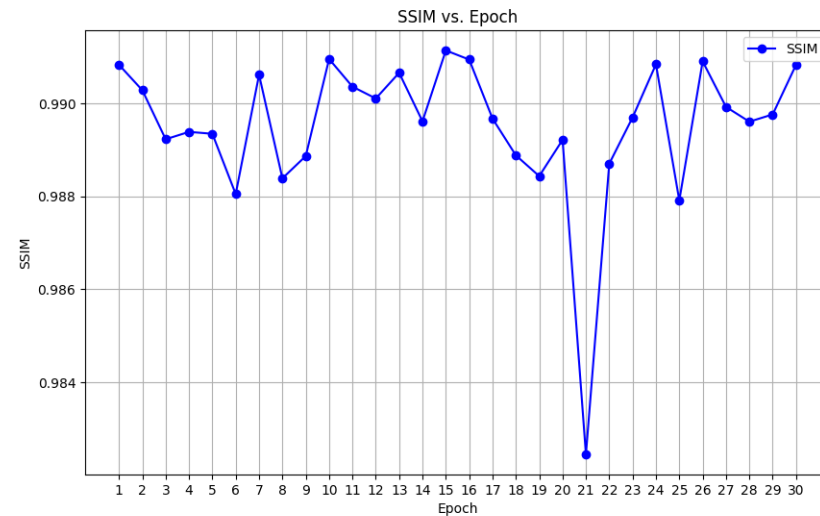
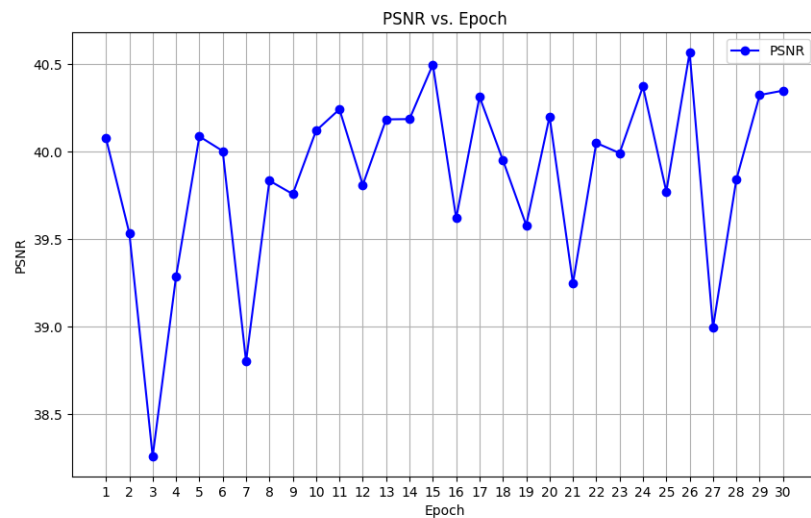
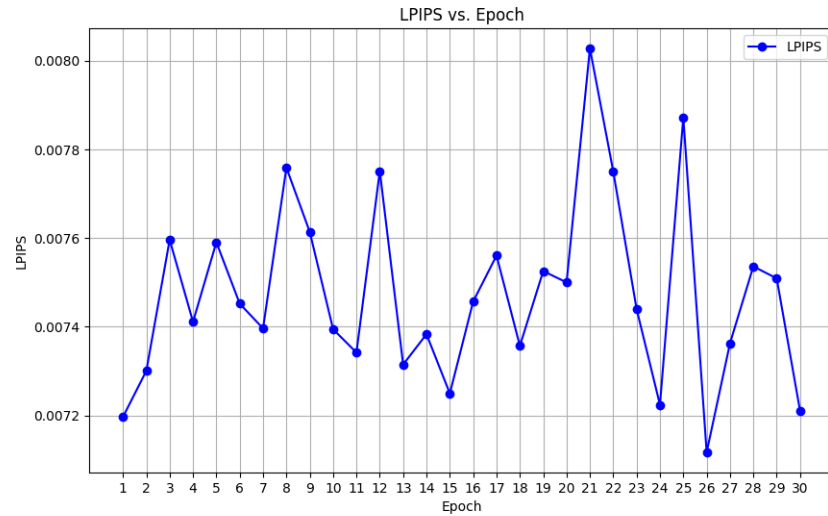
# Metrics Results

## License Plates Dataset



# Metrics Results

**Pretrain:** License Plates Dataset  
**Fine Tune:** License Plates Dataset

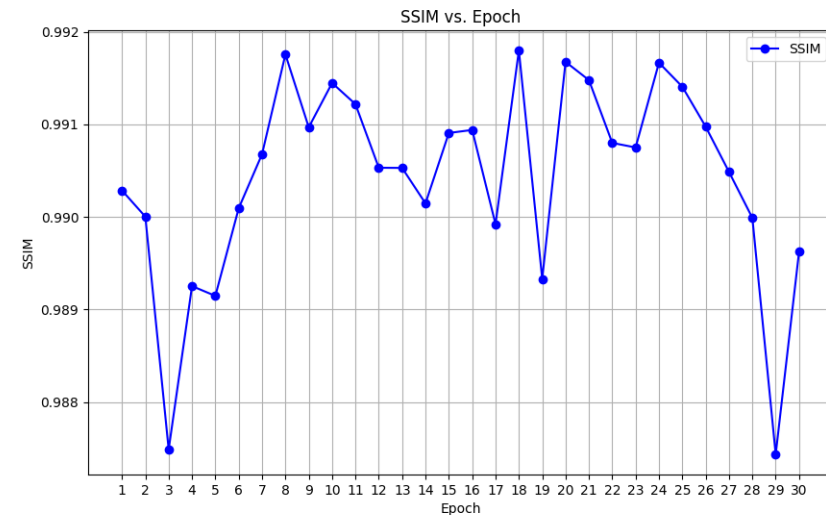
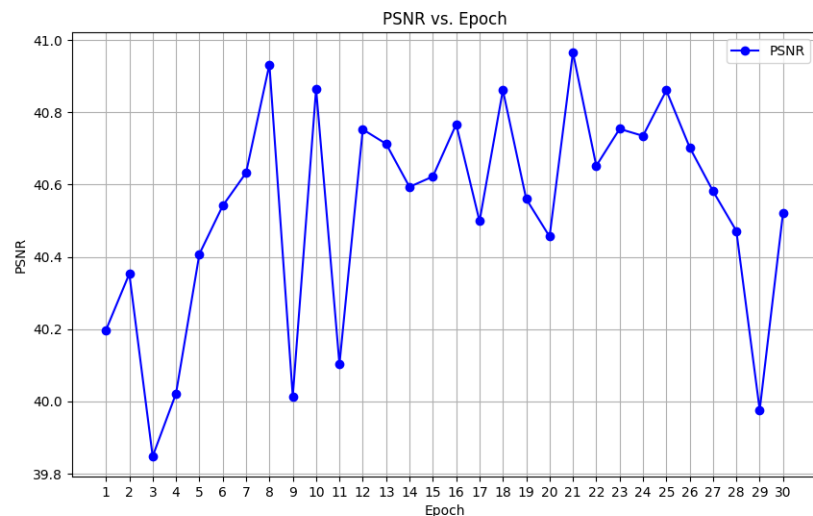
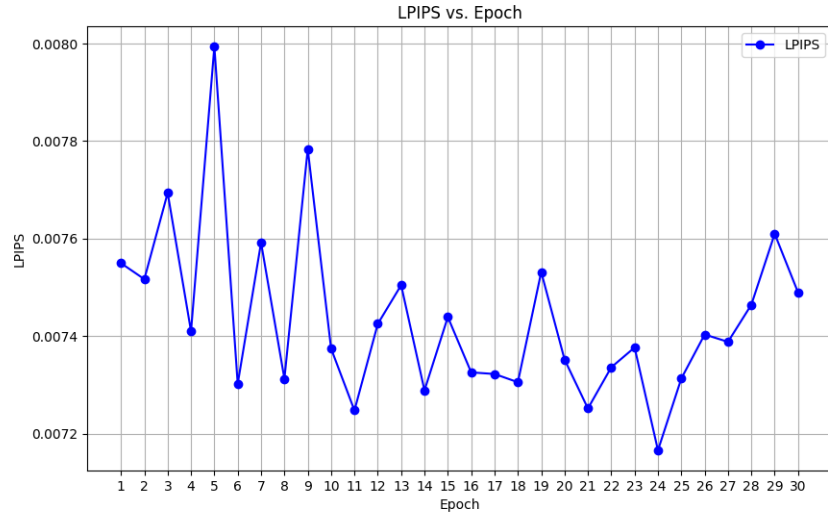




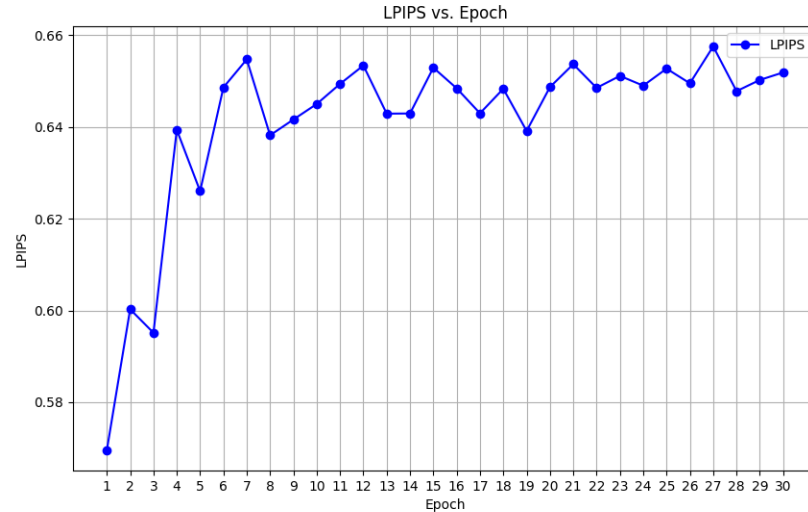
# Metrics Results

**Pretrain:** License Plates Dataset + DIV2K

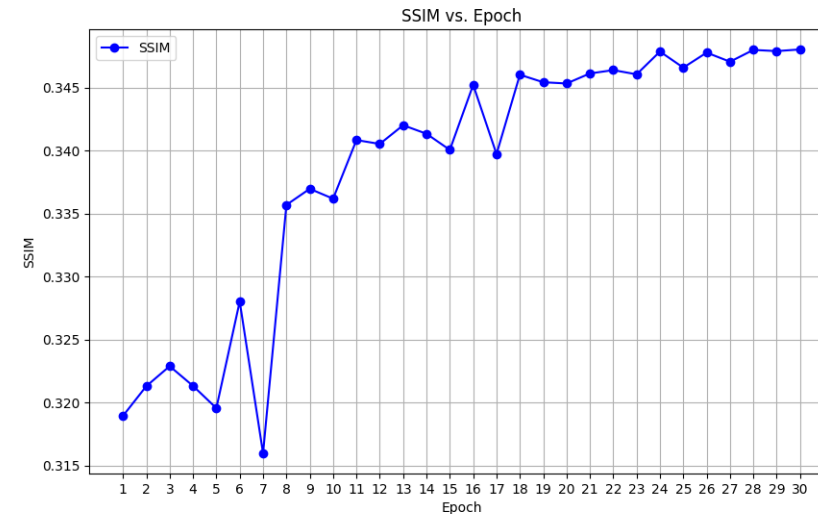
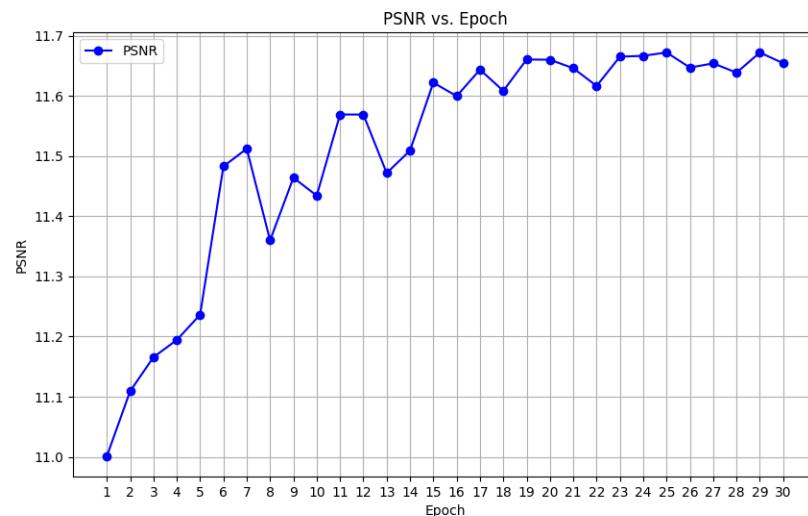
**Fine Tune:** License Plates Dataset



# Metrics Results



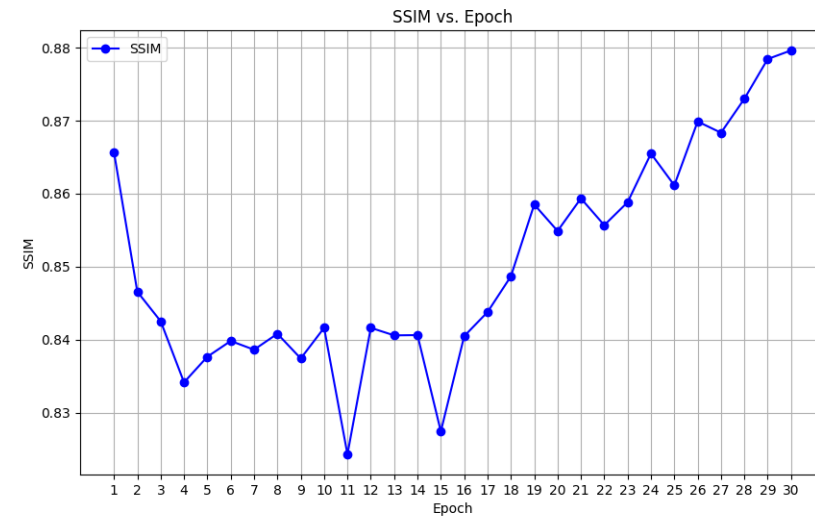
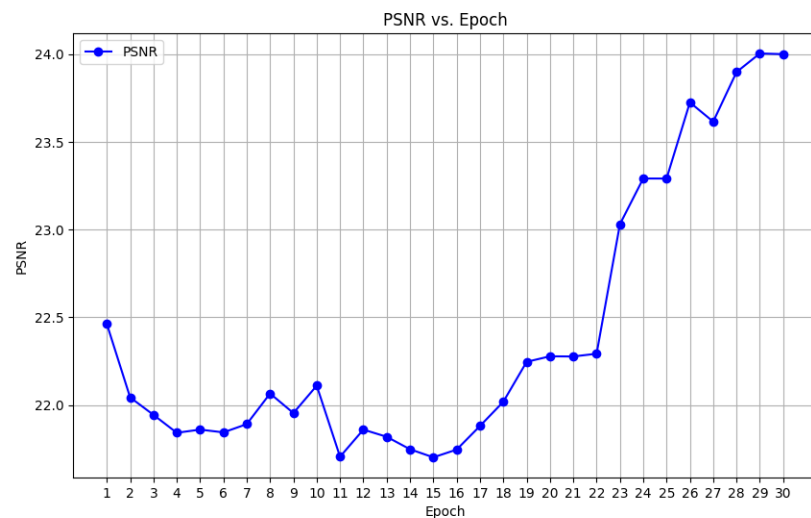
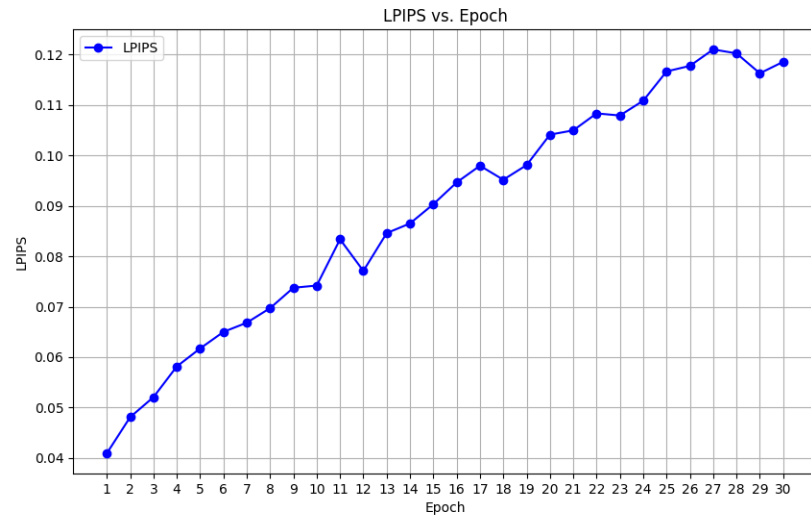
**Pretrain:** License Plates Dataset  
**Fine Tune:** License Plates Dataset  
Distorted








# Metrics Results

**Pretrain:** License Plates Dataset + DIV2K

**Fine Tune:** License Plates Dataset Distorted



# The Results + Best Model

<u>Experiment</u>	<u>Distortion</u>	<u>Model</u>	<u>PSNR</u> ↑	<u>SSIM</u> ↑	<u>LPIPS</u> ↓	<u>Result Output</u>
1	X	Our Model	38	0.985	0.013	
2	X	Our Model	40.4	0.992	0.0072	
3	V	Our Model	11.68	0.348	0.657	
4	V	Our Model	24	0.88	0.119	
5	X	Article Model	25.95	0.735	0.255	



# Model Results

SR method	Cyclic Path	Network structure	PSNR↑	SSIM↑	LPIPS↓
SRResCycGAN	×	$\mathbf{y} \rightarrow \mathbf{G}_{\text{SR}} \rightarrow \hat{\mathbf{y}}$	25.05	0.67	<b>0.3357</b>
SRResCycGAN	✓	$\mathbf{y} \rightarrow \mathbf{G}_{\text{SR}} \rightarrow \hat{\mathbf{y}} \rightarrow \mathbf{G}_{\text{LR}} \rightarrow \mathbf{y}'$	26.13	0.71	0.3911
SRResCycGAN+	✓	$\mathbf{y} \rightarrow \mathbf{G}_{\text{SR}} \rightarrow \hat{\mathbf{y}} \rightarrow \mathbf{G}_{\text{LR}} \rightarrow \mathbf{y}'$	<b>26.39</b>	<b>0.73</b>	0.4245

SR methods	#Params	sensor noise ( $\sigma = 8$ )			compression artifacts ( $q = 30$ )		
		PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
EDSR [13]	43M	24.48	0.53	0.6800	23.75	0.62	0.5400
ESRGAN [22]	16.7M	17.39	0.19	0.9400	22.43	0.58	0.5300
ESRGAN-FT [15]	16.7M	22.42	0.55	0.3645	22.80	0.57	0.3729
ESRGAN-FS [5]	16.7M	22.52	0.52	0.3300	20.39	0.50	0.4200
SRResCGAN [18]	380K	25.46	0.67	0.3604	23.34	0.59	0.4431
SRResCycGAN (ours)	380K	25.98	0.70	0.4167	23.96	0.63	0.4841
SRResCycGAN+ (ours)	380K	26.27	0.72	0.4542	24.05	0.64	0.5192
	unknown corruptions [17]						
SRResCGAN [18]	380K	25.05	0.67	0.3357			
SRResCycGAN (ours)	380K	26.13	0.71	0.3911			
SRResCycGAN+ (ours)	380K	26.39	0.73	0.4245			
	real image corruptions [24]						
SRResCycGAN (ours, valset)	380K	28.6239	0.8250	-			
SRResCycGAN (ours, testset)	380K	28.6185	0.8314	-			

# Conclusions

- **Pretraining on DIV2K :**

Improved model performance ,  
enabling better generalization and higher quality outputs.

- **Dataset Size:**

Larger training datasets led to superior model results ,  
enhancing robustness and image fidelity.

- **Distortion Handling :**

Training with distortions made the model not effective  
at denoising but still effective at reconstructing high-resolution images  
from low resolution input.

- **Optimal Configuration :**

Combining pretraining, a larger dataset ,  
produced the best overall performance in the SRcycGAN model.

# Discussions and Future Work

## Discussions:

### •**Model Flexibility:**

The **SRcycGAN** model worked, but its performance changed depending on the image resolution that was trained. The model worked well on **bicubic distortion** but not on dimensional distortion.

### •**Generalization:**

Pretraining helped improve results, but the model's ability to work on new data still depends on how similar the training data dimensions are

## Future Work:

•**Test with More Datasets:** Use a wider variety of datasets to see how well the model works across bigger number of images.

•**Add More Distortions:** Introduce more complex distortions to make the model better at handling real-world scenarios.

# Final Summary

- We need to learn how to build models that efficiently use resources, ensuring that we get the best performance without unnecessary computation complexity.
- We need to understand how to better select the appropriate data that aligns with our specific goals, which is crucial for achieving accurate results.
- We need to develop better skills in cleaning data effectively, making sure that it's ready for the next steps in the modeling process.
- We have explored additional techniques for refining data and handling various distortions in GAN networks, which will help in producing more reliable and robust models.





**Thank You!**

**Only together we shall win!**

